Tuomas Louhelainen

# CareMe-opetuspelin kehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

4.5.2015

Metropolia

| | |
|---|---|
| Tekijä(t)<br>Otsikko<br><br>Sivumäärä<br>Aika | Tuomas Louhelainen<br>CareMe–opetuspelin kehitys<br><br>30 sivua + 1 liite<br>4.5.2015 |
| Tutkinto | Insinööri (AMK) |
| Koulutusohjelma | Tietotekniikka |
| Suuntautumisvaihtoehto | Ohjelmistotekniikka |
| Ohjaaja(t) | Lehtori Miikka Mäki-Uuro<br>Lehtori Antti Laiho |


Tarkoituksena insinöörityössä oli jatkokehittää ja hioa jo aloitettua pelikonseptia ja tuottaa siitä toimiva demoversio, jota voidaan käyttää laajempaan käyttäjätestaukseen. Tästä versiosta olisi mahdollista kehittää tuotantoversio.

Insinöörityössä esiteltiin CareMe:ssä käytettyjä työkaluja ja tekniikoita niin asiakasohjelmisto- kuin pilvipalvelupuolella. Työssä käytiin läpi pelin toimintaa käyttäjätasolla ja tutustuttiin pelin taustajärjestelmien logiikkaan.

Työn alkupuolella tutustutaan käytettäviin työkaluihin, ohjelmistoihin ja palveluihin. Työkalujen jälkeen käydään läpi pelin logiikkaa yleisesti. Seuraavaksi perehdytään pelin kehitykseen ja eri järjestelmien integrointiin yhdeksi toimivaksi kokonaisuudeksi.

Työssä tutustutaan laajasti Amazon Web Service –pilvipalveluratkaisuihin ja niiden käyttöön Unity-peliprojektin taustajärjestelminä. Tämän lisäksi tutustutaan web-sovelluksen autentikointi- ja suojaustekniikoihin perehtymällä RSA- ja AES-salauksiin.

Insinöörityön lopputuloksena syntyi toimiva demo, jota voidaan edelleen jatkokehittää valmiiksi tuotteeksi. Demo sisältää yhden ympäristön ja yhden animoidun potilaan. Käyttäjien autentikointi- ja taustajärjestelmät kehitettiin käyttäjätestauksen mahdollistavalle tasolle. Skenaarioeditorilla pystyy luomaan omia potilastapauksia ja tallentamaan ne pilveen muiden käyttäjien saataville.



| | |
|---|---|
| Avainsanat | CareMe, Unity, AWS, pilvipalvelu, oppimispeli, simulaatio |

| | |
|---|---|
| Author(s)<br>Title<br><br>Number of Pages<br>Date | Tuomas Louhelainen<br>Development of CareMe Learning Game<br><br>30 pages + 1 appendices<br>4.5.4.2015 |
| Degree | Bachelor of Engineering |
| Degree Programme | Information Engineering |
| Specialisation option | Software Engineering |
| Instructor(s) | Miikka Mäki-Uuro, Senior Lecturer<br>Antti Laiho, Senior Lecturer |

The purpose of the thesis was to continue developing the CareMe-simulation game concept and produce a functional demo version that could be used for a wider user testing. The final product could then be further developed to the production version stage.

The thesis presents tools and techniques used in developing CareMe. The thesis covers both the Cloud and Client side development.

The first part describes the technologies, software and services used. After this, some basic level of game side logic is described from the user's perspective. Finally, the study explains the developing of the game and integrating multiple systems and services into one working product.

The thesis focuses extensively in the Amazon Web Service – Cloud service and us it as part of the background systems of the Unity game project. Additionally, the study explores the authentication and security technologies used in web applications by studying the RSA and AES encryptions.

The result of the thesis was a functional demo, which could be then developed further into a final product. The demo includes one hospital environment and one animated patient model. Users are able to create their own patient scenarios using build in editor and save them directly into the Amazon Cloud.

| | |
|---|---|
| Keywords | CareMe, Unity, AWS, Cloud service, learning game, simulation |

## Sisällys

## Lyhenteet

| | |
|---|---|
| AES | *Advanced Encryption Standard*. Joan Daemenin ja Vincent Rijmenin kehittämä salausmenetelmä. |
| AWS | *Amazon Web Services.* Kokoelma on demand –pilvipalveluresursseja. |
| CRUD | *Create, read, update, delete.* Käyttöliittymäkonsepti. |
| DBaaS | *Database as a Service.* Itsenäisenä pilvipalveluna toimiva tietokanta. |
| EC2 | *Elastic Cloud Computing.* AWS:een kuuluva virtuaalipalvelinjärjestelmä. |
| REST | *Representational state transfer.* Http-protokollaan perustuva arkkitehtuurimalli. |
| RSA | Julkisen avaimen salausmalli. Nimi tulee kehittäjien nimistä Rivest, Shamir ja Adleman. |
| S3 | *Simple Storage Service.* AWS:een kuuluva tiedontallennusjärjestelmä. |
| SES | *Simple Email Service.* AWS:een kuuluva sähköpostijärjestelmä. |
| SQL | *Structured Query Language*. Relaatiotietokannoissa käytettävä kyselykieli. |
| WYSIWYG | *What You See Is What You Get.* Ohjelmisto jossa sisältö näyttää muokatessa samalta kuin lopputulos. |

# 1 Johdanto

Tämän insinöörityön tavoitteena oli jatkaa Unity3d:llä kehitettyä CareMe-oppimispeliprojektia ja saada aikaan toimiva demo jota voitaisiin kehittää edelleen lopulliseksi tuotteeksi opetuskäyttöön.

Työssä tutustutaan kehittämiseen käytettyihin työkaluihin ja käydään läpi itse pelin toimintaa pelaajan näkökulmasta. Työssä perehdytään myös pelin taustajärjestelmien tekniikkaan ja toimintaan.

CareMe on virtuaalinen oppimispeli, jossa pelaaja ratkaisee asiantuntijan tekemiä potilasskenaarioita erilaisissa virtuaalisissa ympäristöissä. Myös muunlaisia ympäristöjä, esim. kotiympäristö, on mahdollista rakentaa peliin. Skenaariot ovat asiantuntijoiden rakentamia ja tukevat opetettavaa opetuskokonaisuutta. Opettaja voi rakentaa peliin hyvinkin erilaisia potilasskenaarioita ja hoitoprosesseja opiskelijan pelattavaksi.

*Pelaaminen* on nykyaikaa. Se aloitetaan tekniikan kehittyessä yhä nuorempana. Nykyään miltei jokaisesta länsimaalaisesta perheestä löytyy mobiililaite, jolla voi pelata. Nuoret oppivat hyvin varhaisessa vaiheessa käyttämään tietoteknisiä laitteita, ja yleensä tämä tapahtuu pelien kautta. Ei ole erikoista nähdä alle 5-vuotias käyttämässä tablettia. Ja mitä tämä tekee? Hän pelaa.

Vuonna 2013 yli puolet 15-24 vuotiaista nuorista pelasi videopelejä aktiivisesti. Vaikka pelaamisen suosio tippuu iän kasvaessa, tippuminen ei ole radikaali. Vanhukset pois lukien karkeasti hieman alle puolet ihmisistä pelaa. Yllättäen yli 45-vuotiaista löytyy paljon pelaajia; naisista jopa 41 % pelaa aktiivisesti. [1.]

*Miksi ihmiset pelaavat niin paljon?* Hyvin kehitetty ja rakennettu peli on palkitseva. Aivot joutuvat työskentelemään ja pysyvät virkeinä. Pelit aktivoivat aivoja erilaisilla pulmilla ja visuaalisilla ärsykkeillä. Pelatessa, ja varsinkin pelissä onnistuessa aivot vapauttavat dopamiinia, joka tuottaa pelaajalle mielihyvää ja tehostaa myös oppimista [2].

Pelit aktivoivat myös tunnepohjaista oppimista. Asiat opitaan helpommin kun niihin liitetään voimakkaita tunteita. Koska nuoret pelaavat jo lapsesta pitäen miltei päivittäin vapaa-ajalla niin miksei pelaamista käytettäisi opetuksessa yhtenä tehokkaana työkaluna?

Insinöörityön alussa tutustutaan CareMe:n taustaan ja projektin tavoitteisiin. Seuraavaksi käydään läpi projektissa käytettäviä työkaluja ja palveluita. Työn keskivaiheilla perehdytään CareMe:n toimintaan käyttäjän näkökulmasta, jonka jälkeen siirrytään tekniseen toteutukseen ja taustajärjestelmiin. Lopussa pohditaan jatkokehitystä ja kootaan yhteen projektista saatuja tuloksia.

# 2 Tausta ja tavoitteet

Taustaluvussa käydään ensin läpi CareMe:n taustaa sekä tutustutaan simulaatio-opetuksen pedagogiikkaan. Tavoiteosiossa tutkitaan projektin alkuasetelmia ja pohditaan projektissa tavoiteltavia tuloksia. Miten pelin eri osa-alueita tulisi kehittää?

## 2.1 Tausta

CareMe:n taustat ovat simulaatio-opetuksessa. Simulaatio-opetuksella tarkoitetaan todellisuuden jäljittelemistä opetustilanteessa. Simulaatio-opetuksessa oppimisen apuna käytetään oikeaa tilannetta simuloivia apuvälineitä, usein simulaatio-nukkea, sekä autenttista potilastapausta. Simulaatio-opetusta on käytetty jo 1950-luvulta lähtien mm. lento-opetuksessa.

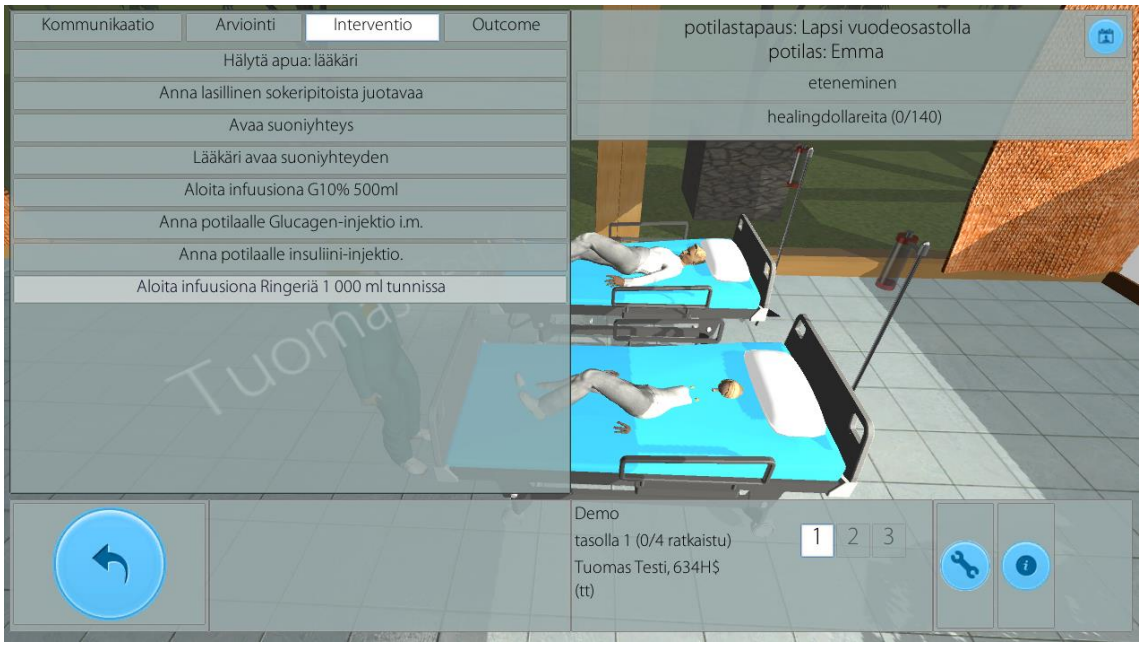
Nykypäivän oppijat suosivat todentuntuisia ympäristöjä. Simulaatio-opetuksessa harjoitellaan oikean elämän tilanteita kokonaisuuksina; Miten toimitaan tällaisessa tilanteessa? Mitkä asiat tulee muistaa tehdä tässä tapauksessa? Simulaatio-opetus mahdollistaa myös tilanteet, joissa harjoitusta voidaan toteuttaa autenttisessa ympäristössä.

Simulaatio-opetus koostuu osista. Ensin asetetaan oppimistavoitteet eli kerrotaan oppijalle harjoituksen tärkeimmät pedagogisesti asiat oppimisen kannalta. Tämän jälkeen suoritetaan itse harjoitus. Harjoitukseen voi osallistua useampi opiskelija samaan aikaan. Ohjaajia simulaatiossa on yleensä yhdestä kahteen. Ohjaajat ohjaavat ja tarkkailevat oppilaita esimerkiksi mikrofonien ja kameroiden välityksellä. Harjoituksen jälkeen pidetään opetuksen kannalta tärkein osuus, palautekeskustelu. Palautekeskustelussa käydään läpi vaihe vaiheelta skenaario ja pureudutaan haastaviin osuuksiin. [3.]

Peli on tarkoitettu terveysalan oppimistilanteisiin. Oppilaitoksille peli antaa mahdollisuuden tarkistella ja pitää yllä oppilaidensa hoitoprosessitietämystä sekä opettaa mm. käytännön harjoitteluun liittyviä asioita etukäteen.

Pelaajat saavat välitöntä palautetta suorituksestaan ja toiminnastaan. Skenaarion lopuksi peli generoi pelaajalle loppuraportin, joka sisältää statistiikkaa skenaarioon ja siinä kehittymiseen liittyen. Myös opettaja pystyy tarkistelemaan näitä tietoja.

CareMe:n kehittäminen aloitettiin vuonna 2009 Metropolian Terveys ja hoitaminen -yksikössä lehtori Jaana-Maija Koiviston vetämänä. Tarpeiden kartoituksen ja pelin suunnittelun jälkeen pelin tekninen puoli toteutettiin alihankintana ulkopuolisella yrityksellä. Pelistä kehitettiin toimiva demoversio (kuva 1) jota testautettiin laajasti Metropolian sisällä terveydenhuollon opiskelijoilla.




Kuva 1. Lähtökohta syksyllä 2014

Syksyllä 2014 projektin kehitys siirrettiin kokonaan Metropolian sisälle. Metropolian soveltavan elektroniikan tutkimus- ja kehitysyksikköön Electriaan kerättiin tiimi pelikehitykseen erikoistuneita opiskelijoita. Tiimiin tuli 3d-artisti viestinnän koulutusohjelmasta, kaksi teollista suunnittelijaa sekä allekirjoittanut, ohjelmistotekniikan opiskelija. Tiiminvetäjänä ja pedagogiikan ja simulaatio-opetuksen asiantuntijana jatkoi lehtori Jaana-Maija Koivisto.

## 2.2 Tavoitteet

CareMe:n aikaisempaa versiota oli laajasti testattu hoitotyön opiskelijoina ja pelkästään sen pelattavauudestakin toteutettiin opinnäytetyö. Työn tuloksista ilmeni, että peli oppimismuotona on toimiva, mutta itse toteutus ontui vielä. Animaatiot eivät vastanneet nykystandardeja ja käyttöliittymä oli monimutkainen. [4.]

Hoitoprosessit kehittyvät, määritykset muuttuvat ja työelämässä eri työpaikoilla on omat tapansa hoitaa eri asiat. Koska hoitoala on alati kehittyvä, pelin yhtenä suurena valttikorttina on mahdollisuus luoda sisältöä itse. Vanhassa versiossa sisällönluonti oli kuitenkin opettajista monimutkaista ja rajoitti liikaa.

Projektin alkaessa CareMe:tä ajettiin Metropolian omalla palvelimella. Palvelimella ajettiin Apache-palvelinohjelmistoa joka tarjoili Play Frameworkin avulla html-sivuja. Tietokantana oli samalla palvelimella toimiva PostgreSQL-tietokanta. Skenaarioeditori oli toteutettu CRUD-tyyppisenä web-lomakkeella Play Frameworkissä.

Tavoitteena oli siis hioa pelattavuutta ja tuoda virtuaalinen oppimispeli lähelle viihdepeleistä totuttua tasoa. Grafiikka, animaatiot ja käyttöliittymä oli päivitettävä ja taustajärjestelmiin oli tehtävä radikaaleja muutoksia. Tarkoituksena oli saada satoja yhtäaikaisia pelaajia tukeva, pilvipalveluna toimiva oppimisympäristö, johon olisi kuitenkin helppo tehdä räätälöityä sisältöä ilman laajaa teknistä osaamista.

Henkilökohtaisina tavoitteina projektissa oli tutkia ja opiskella uusia teknologioita ja tekniikoita. Erityisen kiinnostavana pidin Amazon Web Service –palveluita ja niiden käyttöä osana Unityllä kehitetyn pelin taustajärjestelmiä.

# 3 Työkalut

## 3.1 Unity

Unity3D on Unity Technologiesin kehittämä kaupallinen pelinkehitysympäristö ja pelimoottori. Se sisältää renderöinti- ja fysiikkamoottorit ja toimii niin 3D- kuin 2D-ympäristöissä. Unity käyttää fysiikkamoottorina Nvidian PhysX –moottoria.

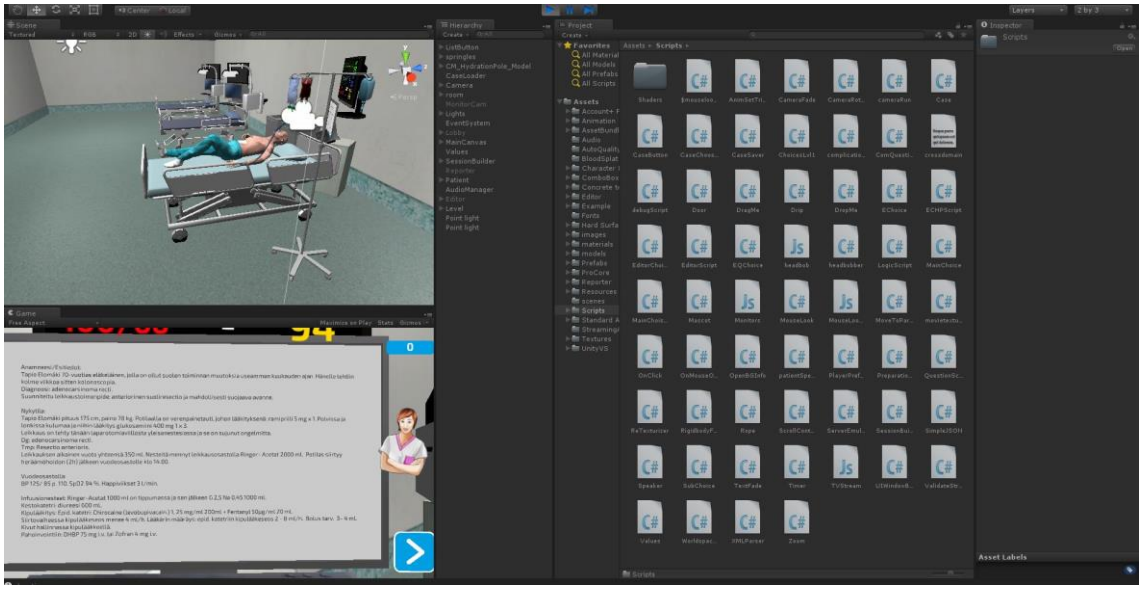
Unityllä kehitetty peli voidaan kääntää helposti eri alustoille. Helmikuussa 2015 Unityn tukemiin alustoihin kuuluu mm. Windows, Mac, Linux, web-selaimet sekä useat mobiilikäyttöjärjestelmät. [5.]

Monialustaisuus oli yksi tärkeä syy Unityn käyttöön projektissa. Tavoitteena oli kehittää peli, jota pystyisi pelaamaan niin koulussa, töissä kuin kotonakin. Selaimissa toimiva Unity Web Player mahdollistaa tämän.

Unity tukee natiivisti kolmea eri ohjelmointikieltä: C#:tä, JavaScriptia ja Boota. Kuten suurimmassa osassa Unity-projekteja, tässä projektissa kielenä oli C#. Unityyn on saatavissa satoja lisäosia niin editoriin kuin pelimoottoriin. Projektissa käytettiin mm. Visual Studion käytön mahdollistavaa UnityVS-lisäosaa. C# kielellä ohjelmoiminen Visual Studiolla onnistuu mielestäni paljon paremmin kuin Unityn oletuksena tarjoamalla MonoDevelop-kehitysohjelmistolla.

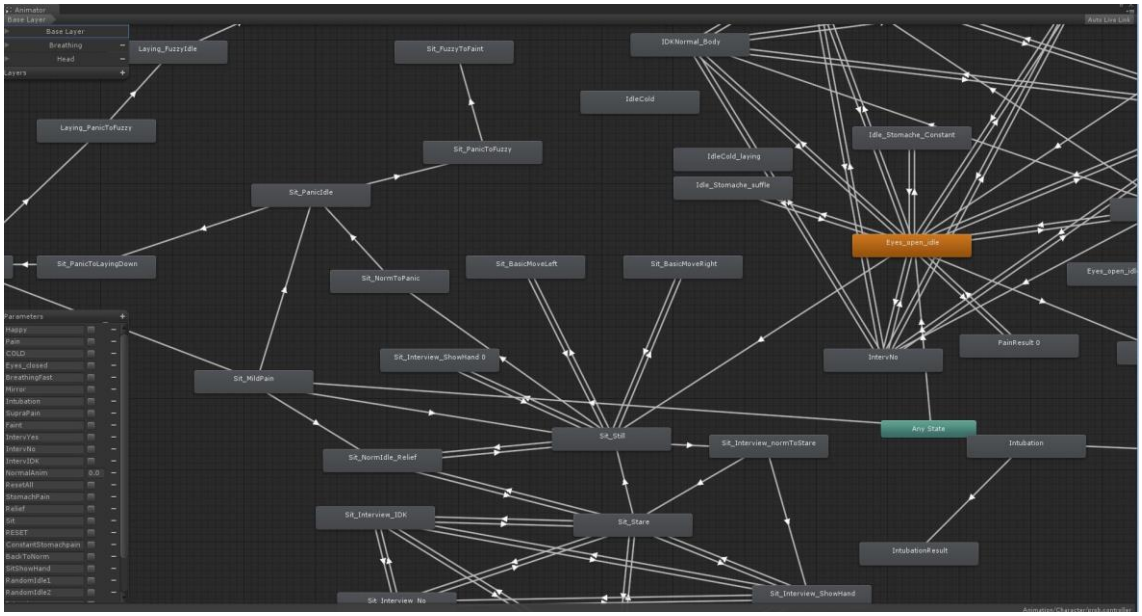
Unity Technologies ylläpitää Unity Asset Storea, jossa 3d-mallintajat, ohjelmoijat ja muut pelialan ammattilaiset myyvät tuotoksiaan. Palvelusta voi ostaa kaikkea aina pienistä koodinpätkistä kokonaisiin projekteihin asti. Kaupasta ostettuja paketteja voi tuoda suoraan Unityyn.

Pelin kehitys Unityllä on intuitiivista. Eri elementtejä voidaan vetää suoraan projektinäkymään. Unity osaa itse lukea useita tiedostomuotoja ja osaa myös käyttää ulkopuolisia ohjelmistoja, kuten 3d-mallinnusohjelmisto Blenderiä, vieraiden tiedostotyyppien lukemiseen. Elementtejä asetellaan projektista riippuen 2d- tai 3d-maailmaan scene-editorissa. Tuloksen näkee suoraan pelinäkymästä. Graafinen käyttöliittymä toimii miltei WYSIWYG-periaatteella. Kun teet muutoksen, näet tuloksen heti (kuva 2).



Kuva 2. Unityn kehitysnäkymä. Peli-, scene-, hierarkia- ja projektinäkymät.

Unityssä hahmoanimaatioon voidaan käyttää Mecanim–animaatiojärjestelmää. Sen hyviä ominaisuuksia ovat helppo työnkulku, yksikertainen animaatioiden yhdistely, useiden animaatiotasojen käyttö päällekkäin, animaatiokurvien käyttö, animaatioiden esikatselu ja suoraviivainen visuaalinen animaatiokarttojen rakennus (kuva 3). Mecanimin avulla on myös helppo siirtää yhdessä ihmismallissa toimivat animaatiot toiseen malliin ilman mallikohtaista kustomointia. [6.]



Kuva 3. CareMe:n potilaan visuaalinen animaatiokartta

## 3.2 Play Framework

CareMe:n taustajärjestelmät on toteutettu Play Framework –sovelluskehyksen ympärille. Play Framework on Javaa ja Scalaa tukeva web-sovelluskehys, joka mahdollistaa modernien, valmiiden komponenttien modulaarisen käytön web-pohjaisissa sovelluksissa [7].

Play Framework perustuu ReST-arkkitehtuuriin. ReST (Representational State Transfer) on arkkitehtuurityyli, joka mahdollistaa hyvinkin erilaisten järjestelmien kommunikaation ja integraation yhteisellä http-kyselyihin perustuvalla rajapinnalla.

## 3.3 Amazon Web Services

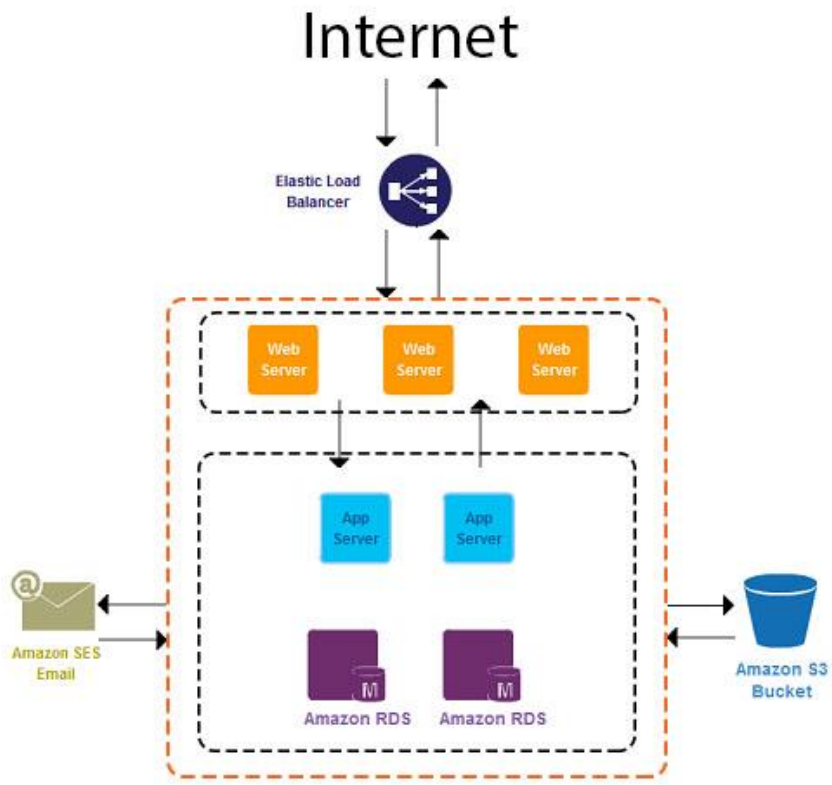
Amazon Web Services, eli AWS, on Amazon Web Services Inc:in tarjoama kokoelma erilaisia on demand -palveluita, jotka yhdessä muodostavat massiivisen pilvipalvelun. [8.]

*Amazon Elastic Compute Cloud EC2* on AWS:n perusta, jonka päälle muut palvelut rakentuvat. EC2 on virtuaalipalvelinpalvelu josta asiakas voi ostaa tuntiveloituksella oman virtuaalipalvelimen. Palvelimia ajetaan instansseina, joita voidaan käynnistää ja sulkea lennosta ja niistä maksetaan vain niiden käynnissäoloajasta.

EC2 tarjoaa eri kokoonpanoilla olevia palvelimia eri tarkoituksiin. Hinta määräytyy koneen teknisten määritysten mukaan: enemmän tehoa, tilaa tai muistia maksaa asiakkaalle enemmän. Asiakas voi valita EC2-palvelimiensa fyysisen sijainnin, täten lyhentäen mahdollisia latenssiaikoja. Asiakkaalla on täydet oikeudet omaan palvelimeensa, joten käytettävän käyttöjärjestelmän voi myös itse valita.

Projektissa tärkeimmät pilvipalvelut, kuten autentikointijärjestelmä, on rakennettu EC2-palvelimien päälle (kuva 4).

*Amazon Simple Storage Service (S3)* on pilvitallennusratkaisu, jossa asiakas maksaa käytetystä tilasta ja datan siirrosta. S3 keskustelee EC2-palvelimien kanssa helposti ja datan siirto näiden välillä on maksutonta. Kuten muissakin AWS:n palveluissa, S3:n fyysisen sijainnin voi asiakas valita.




Kuva 4. CareMe:n taustajärjestelmien rakenne

*Amazon Simple Email System (SES)* tarjoaa muiden AWS:n palveluiden kanssa yhteensopivan skaalautuvan sähköpostitusjärjestelmän. SES:iä käyttäessä asiakas maksaa vain lähetetyistä viesteistä ja niiden liitteistä.

*Amazon Relational Database System (RDS)* on EC2-palvelimien päälle rakennettu, helposti skaalattavissa oleva DBaaS-periaatteella toimiva SQL-tietokantajärjestelmä. RDS tukemiin tietokantoihin kuuluu mm. projektissa käytetty MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server (2008 R2 sekä 2012) sekä MySQL-yhteensopiva Amazon Aurora. Kuten muissakin AWS:n kuuluvista palveluista, RDS:n hinta muodostuu käytettävistä resursseista.

*Amazon Elastic Load Balancing* hoitaa sovellusliikenteen kuormantasauksen eri palvelinten kesken. Se myös mahdollistaa suuremman saatavuuden palvelulle; jos yksi palvelimista kaatuu, Elastic Load Balancing ohjaa liikenteen muille palvelimille.

## 3.4 Tietokannat

*MongoDB* on MongoDB Inc:in ylläpitämä, avoimen lähdekoodin no-SQL–tietokanta. Se mahdollistaa erittäin skaalautuvien, isojen tietokantojen nopean käsittelyn ja replikoinnin [9].

MongoDB:ssä voi käyttää jopa 1000 palvelinta rinnakkain. Niistä valitaan satunnaisesti yksi palvelin toimimaan isäntänä, jolta kaikki muut kopioivat tiedot. Jos yksi palvelin rikkoutuu, tiedot ovat muilla palvelimilla turvassa. Kun taas palvelinverkkoon lisätään uusi palvelin, kopioi se muiden palvelinten tiedot itselleen. Prosessi on hyvin samankaltainen kuin RAID-1-kovalevyissä [10].

MongoDB eroaa muista yleisesti käytöissä olevista tietokannoista oman kyselykielensä takia. Se ei ole relaatiotietokanta, vaan linkitykset toteutetaan ohjelmistotasolla.

CareMe-projektiin MongoDB valittiin potilastapausten tietokannaksi juuri no-SQL-luonteensa takia. Tietokantaan voidaan tallentaa isojakin tietokokonaisuuksia, kuten pelin potilastapauksia.

*MySQL* on alun perin suomalisen Michael ”Monty” Wideniuksen ja ruotsalaisen David Axmarkin perustaman MySQL AB:n kehittämä relaatiotietokantaohjelmisto. Ohjelmisto on nimetty Wideniuksen My-nimisen tyttären mukaan. Tällä hetkellä MySQL:n omistaa Oracle Corporation. CareMe:ssä MySQL–tietokantaa käytetään käyttäjätietojen tallennukseen ja hallintaan.

MySQL-relaatiotietokanta koostuu monista tietokannoista. Nämä tietokannat pitävät sisällään tauluja. Taulut rakentuvat kentistä ja riveistä. MySQL-taulua voisi verrata Excel-tauluun. Eri taulujen kentät voivat olla relaatiosuhteessa toisiinsa [11]. CareMe:ssä eri tauluista haetaan mm. eri ryhmien ja skenaarioiden välisiä suhteita.

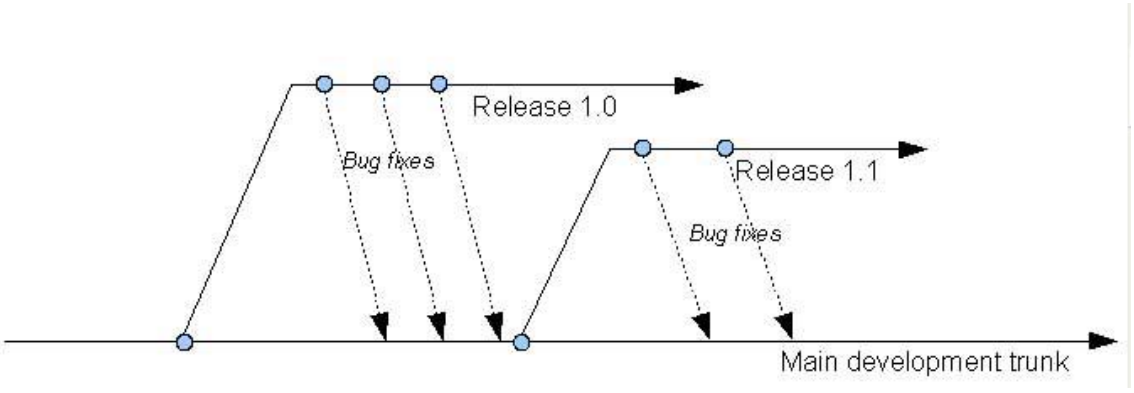
## 3.5 Apache

Apache HTTP Server on Apache Software Foundationin kehittämä ja ylläpitämä avointa lähdekoodia käyttävä, ilmainen web-palvelinohjelma. Vuonna 2010 Apachea käytettiin yli 120 miljoonassa palvelimessa. Tälläkin hetkellä yli puolet internetsivuista jaetaan Apachea käyttäen [12]. Apachen avulla voidaan myös ajaa PHP- ja muita web-sovelluksiin liittyviä ohjelmakoodeja.

## 3.6 Subversion

Projektin versionhallintajärjestelmänä käytettiin Apache Subversionia. Subversionin versioarkiston tarkasteluun käytimme TortoiseSVN-asiakasohjelmaa. Subversion, lyhennettynä svn, on Apache Foundationin kehittämä ja ylläpitämä yhden keskuspalvelimen käyttöön perustuva versionhallintajärjestelmä. Subversion valittiin projektin versionhallinnaksi saatavilla olevan Metropolian palvelimen takia. Subversionin isojen binääritiedostojen parempi hallinta ja yhden keskuspalvelimen logiikka oli pääetuna sen vaihtoehtoihin, esim. GIT:iin verrattuna [13].

SVN-versiohallinnassa projektinkehitys jaetaan puumaiseen rakenteeseen. Puun runko, eli päähaara pitää sisällään aina toimivan version. Kun uutta versiota lähdetään kehittämään, ajetaan tiedostot versionhallintaan uutena haarana. Kun haara on toimiva ja erinäiset bugit on korjattu, yhdistetään haara runkoon. Näin pystytään pitämään versiohallinnassa aina uusinta toimivaa versiota (kuva 5).




Kuva 5. SVN-versiohallinnan runko ja haarat [14].

# 4 CareMe yleisesti

Peliä lähdettiin kehittämään nopeasti Scrum-menetelmällä. Ensimmäinen käyttäjätestaus olisi noin kuukauden päässä kehityksen aloittamisesta.

Ensimmäiset päivät projektissa käytettiin suunnitteluun. Kävimme aivoriihessä läpi vanhan tuotteen hyviä ja huonoja puolia ja peilasimme pelinkehityskokemuksiamme vanhaan peliin.

Tulimme nopeasti yhteisymmärrykseen siitä, että oli tehokkaampaa hylätä vanha versio kokonaan ja luoda kokonaan uusi alusta. Vanha alusta ei kykenisi taipumaan uusiin tarpeisiin. Jotta peli olisi skaalautuvampi ja saatavuusaste nousisi, olisivat taustajärjestelmät siirrettävä dedikoidulta palvelimelta suurempaan pilveen hajautettuna.

Vanhaa versiota oli testautettu paljon Metropolian sairaanhoito-opiskelijoilla ja kävimme heidän kanssa keskusteluja uuden version muutoksista (liite 1). Suurimpia ongelmia olivat pelin interaktiivisuuden puute ja tylsyys. Se ei ollut tarpeeksi pelillinen. Virtuaalisen oppimisen pitäisi olla motivoivaa.

Pelin motivoivuus tarvitsee monen osa-alueen toimivuutta. Pelin on oltava tarpeeksi yksinkertainen ja riittävän helppo, jottei innostus lopahda heti alussa. Pelin on kuitenkin oltava haastava. Ilman haasteita peli muuttuu nopeasti tylsäksi ja puuduttavaksi.

Jotta pelaaja haluaisi kehittyä pelissä, on sen pelaajalle tarjottava palkintoja. Jopa yksinkertainen pistejärjestelmä ja ajoittainen kannustus esim. “hyvin tehty” -huudahduksen muodossa on riittävä tähän.

Jos nämä osa-alueet ovat tasapainossa keskenään, haluaa pelaaja kehittyä. Opetuspelien tapauksessa kehittyminen pelissä tarkoittaa myös kehittymistä opetettavassa aihealueessa. Pelin on siis tarjottava tarpeeksi ohjeistusta ja palautetta pelaajan tukeakseen pelaajan kehittymistä.

## 4.1 Pelin vaiheet

Peli koostuu eri aihealueisiin keskittyvistä potilastapauksista. Hyvässä simulaation potilastapauksessa on vain muutama selkeä oppimistavoite. Se ei myöskään ole liian pitkä.

Peli lukee potilasskenaariot XML-tiedostoista. Tiedostossa määritellään mm. potilastapauksen esitiedot, mahdollisessa monitorissa näkyvät vitaaliarvot, potilaan ulkoasu, potilaan oletusanimaatiot sekä oppimistavoitteet. Tiedostossa on myös määritelty potilastapausten viiden eri osion sisältö (kuva 6).

```
    <SUBCHOICE txt="Tiedätkö missä olet?" score="-20">
        <RESULT>
            <WRONG></WRONG>
            <ANIM>IntervYes</ANIM>
            <CONVERSATION>
                <PATIENT>Joo...</PATIENT>
            </CONVERSATION>
            <MASCOT>Vältä kysymyksiä, joihin voi vastata kyllä tai ei.</MASCOT>
        </RESULT>
    </SUBCHOICE>

    <SUBCHOICE txt="Missä olet?" score="10">
        <RESULT>
            <RIGHT></RIGHT>
            <CONVERSATION>
                <PATIENT>Sairaalan teho-osastolla</PATIENT>
            </CONVERSATION>
        </RESULT>
    </SUBCHOICE>
</INTERVIEW>

<EXAMINATION>
    <SUBCHOICE txt="A - Airway (ilmatie)"  score="5" >
        <RESULT>
            <RIGHT></RIGHT>
            <MASCOT>Potilas on vastannut kysymyksiisi, joten ilmatie on auki.</MASCOT>
        </RESULT>
    </SUBCHOICE>
```

Kuva 6. Osa potilasskenaarion XML-tiedostoa

Potilasskenaario etenee ennalta määrätyn kaavan mukaan. Ensin haastatellaan potilasta. Haastattelussa pyritään arvioimaan tilannetta puhuttamalla potilasta; esim. jos potilaan puheentuotto on haastavaa, voidaan tästä päätellä jotain.

Peruskaavaan voidaan tehdä joitain muutoksia; voidaan esim. jättää haastattelu-osio kokonaan pois, jos potilas on tajuttomana. Kuitenkin perusrakenne (valmistelu, haastattelu, tutkiminen, toimenpide + komplikaatio) on yleisin CareMe:llä rakennetuissa potilasskenaarioissa.

Hoidon tarpeen arviointia jatketaan haastattelun jälkeen tutkimuksilla; potilas tutkitaan usein ABCDE-menetelmän avulla [15]:

- A - Airway
- B - Breathing
- C - Circulation
- D - Disability
- E – Exposure.

Muitakin tämän tyyppisiä rakenteita ja proseduureja, esim. Glasgow'n kooma-asteikko (kuva 7), on kouluttajan erittäin helppo rakentaa pelin skenaarioeditorissa.

| Response | Score |
|---|---|
| Eye opening | |
| Opens eyes spontaneously | 4 |
| Opens eyes in response to speech | 3 |
| Open eyes in response to painful stimulation (eg, endotracheal suctioning) | 2 |
| Does not open eyes in response to any stimulation | 1 |
| Motor response | |
| Follows commands | 6 |
| Makes localized movement in response to painful stimulation | 5 |
| Makes nonpurposeful movement in response to noxious stimulation | 4 |
| Flexes upper extremities/extends lower extremities in response to pain | 3 |
| Extends all extremities in response to pain | 2 |
| Makes no response to noxious stimuli | 1 |
| Verbal response | |
| Is oriented to person, place, and time | 5 |
| Converses, may be confused | 4 |
| Replies with inappropriate words | 3 |
| Makes incomprehensible sounds | 2 |
| Makes no response | 1 |

Kuva 7. Glasgow'n kooma-asteikko [16].

Valmistelu

Valmisteluvaihe on tapauksesta riippuen valinnainen kohta, jossa valmistellaan esitietojen perusteella potilaspaikka. Valmistelussa voidaan opettaa esim. verensiirrossa tarvittavien tutkimusten tilaus lomakkeella (kuva 8).


Kuva 8. Verensiirron valmistelu

Valmistelukohdassa pelaajalle esitetään monta vaihtoehtoa, jotka ovat joko oikein tai väärin. Pelaajan pitää valita näistä mielestään oikeat ja vahvistaa valinta. Fasilitaattori kertoo palautteen, ja valituista vaihtoehdoista oikeat muuttuvat vihreiksi ja väärät punaisiksi. Kohdasta ei pääse eteenpäin ennen kuin kaikki oikeat vaihtoehdot ovat valittuna.

Valmistelukohtien määrä ei ole rajattu. Kohdat voidaan otsikoida eri nimillä ja vaihtoehtoihin voidaan liittää kuvia.

Valmistelu- ja komplikaatiokohdissa voidaan käyttää valintalaatikoissa pelkän tekstin sijasta myös kuvia. Potilastapauksen laatijalla on mahdollisuus ladata kuvat palveluun skenaarioeditorissa erillisen latauslomakkeen kautta.

Haastattelu

Haastattelukohta on yksi pelin kolmesta perusosasta. Useimmissa potilastapauksissa tämä on olennainen osa tapausta, mutta erikoistapauksissa tämäkin voi olla tyhjä, esim. jos potilas on tajuttomana eikä pysty vastaamaan.

Haastattelussa isona ongelma pelimekaniikan suunnittelussa oli se tieto, ettei terveydenhuollossa ole vääriä kysymyksiä. Pitkän pohdinnan ja erilaisten käyttäjätestausten jälkeen päädyimme ratkaisuun, jossa pelaajalla on käytössään rajattu määrä kysymyksiä. Hänen pitää listatuista kysymyksistä valita olennaisimmat tapauksen kannalta. Opiskelijan on opittava priorisoimaan tärkeät kysymykset. Jotta tilanne olisi autenttisen tuntuinen ja interaktiivinen, vastaa potilas kysymykseen puhekuplalla, ääntelyllä ja animaatiolla (kuva 9).


Kuva 9. Potilaan haastattelu

Kun käytettävissä olevien kysymysten määrä tulee täyteen, sulkeutuu haastattelu pelaajalta. Pelaaja voi tarkistaa jo kysytyt kysymykset lokista.

Tutkiminen

Tutkiminen on toinen pelin kolmesta pääkohdasta. Tämä kohta on miltei aina käytössä, mutta voidaan erikoistapauksissa jättää tyhjäksi.

Tutkiminen-kohdassa on listattuna erilaisia tutkimuksia, osa relevantteja ja osa vähemmän relevantteja. Näistä pelaaja valitsee haluamansa tutkimukset. Tutkimusten tulokset voidaan ilmentää pelaajalle erilaisina animaatioina, tekstuurivaihdoksilla ja kameraajoina olennaisiin kohtiin.

Kuitenkaan kaikkia tutkimuksia ja niiden tuloksia ei voida mallintaa virtuaalimaailmassa. Tällaisten tutkimusten ja myös toimenpiteiden tulokset tuodaan käyttäjälle esiin fasilitaattorin kommentteina (kuva 10). Skenaarion luoja voi kirjoittaa editorissa erilaisia kommentteja, perusteluja ja toiminnan tuloksia fasilitaattorin teksteiksi.


Kuva 10. Fasilitaattorin kommentti tutkimustilanteessa

Toimenpiteet

Toimenpiteet-kohta on pelin olennaisin kohta, jossa tapahtuu eniten. Pelaajalle annetaan toimenpidevaihtoehtoja joista hän valitsee olennaiset.

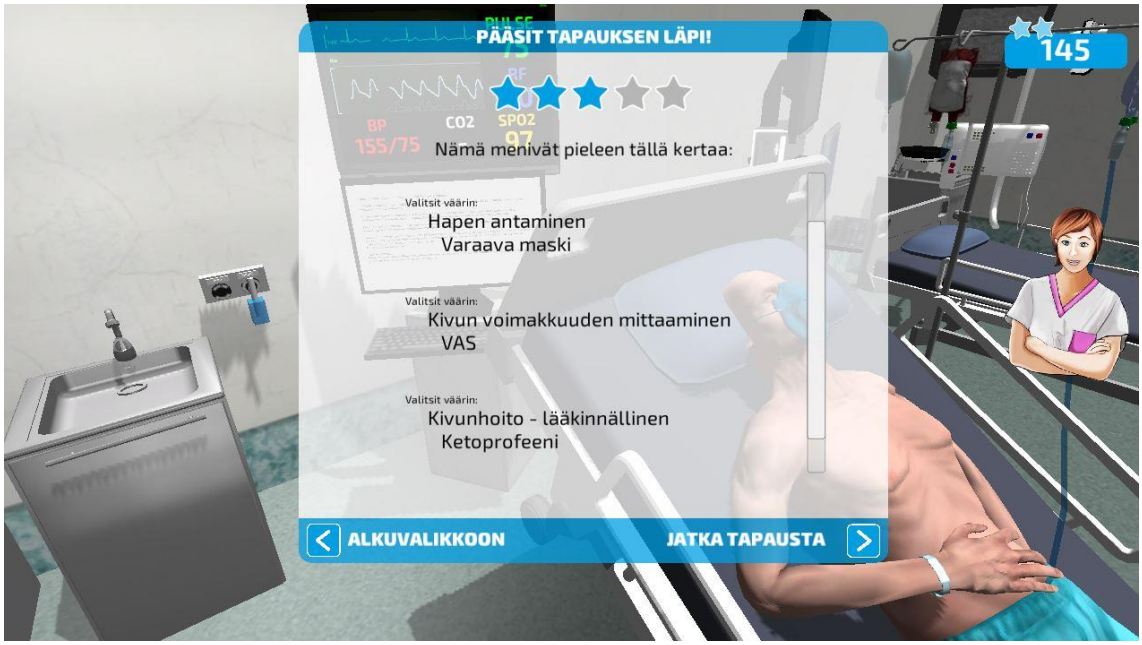
Näissä toimenpiteissä pelaaja joutuu valitsemaan oikeat toimenpidetavat, esim. oikean happimäärän valinta tai tilanteeseen sopivan lääkkeen antamisen. Käyttäjätestausten pohjalta peliin luotiin myös lääkärihahmo, jolta voidaan tietynlaisissa tilanteissa konsultoida mm. lääkeannoksen määrää tai seuraavaa toimenpidettä.

Peli vastaa toimenpiteisiin animoiduilla välineillä, potilaan animaatioilla ja fasilitaattorin kommenteilla (kuva 11).


Kuva 11. Hapen antaminen venturimaskilla neurokirurgiselle potilaalle.

Kun kaikki oleelliset toimenpiteet on suoritettu, peli päättyy. Pelin päättyessä pelaajalle näytetään loppuraportti, jossa listataan kaikki virheet ja oikeat valinnat. Loppuraportissa näkyy myös pelaajan arvostelu tähdillä asteikolla 0-5 (kuva 12).

Tämän tutkielman kirjoitusvaiheessa pelin loppuraportti oli vielä demovaiheessa. Myöhemmissä versioissa loppuraportti tulisi sisältämään graafista statistiikkaa ja tarkempia erittelyjä skenaarion erityisen vaikeista ja helpoista kohdista, sekä historiikkia omasta kehityksestä skenaariossa.




Kuva 12. Loppuraportti

Komplikaatio

Potilastapauksen luoja voi asettaa komplikaation laukeamaan valitsemastaan vaihtoehdosta. Komplikaatio voi laueta niin haastattelusta, tutkimuksesta kuin toimenpiteestäkin. Komplikaatiotilanne muuttaa täysin pelin luonnetta. Sen alkaessa ei olekaan enää aikaa rauhassa lukea tekstejä ja etsiä tietoa kirjoista ja internetistä.

Yleensä komplikaatioon liittyy jokin isompi muutos potilaan tilassa: potilas voi esimerkiksi mennä yhtäkkiä tajuttomaksi, sidottu haava voi alkaa vuotamaan tai potilas voi saada sairaskohtauksen.

Komplikaation alkaessa kameran kuva muuttuu epäselväksi, värit muuttuvat punaiseksi ja kello alkaa laskemaan alaspäin. Peli yrittää simuloida komplikaation liittyvää stressitilaa. Pelaajalle tulee kiire. Oikeassa hoitotilanteessa tulee vastaan tilanteita, joissa ei yksinkertaisesti ole aikaa miettiä pitkiä aikoja, mitä seuraavaksi tutkitaan tai tehdään vaan on tehtävä nopeita päätöksiä.

Komplikaatiotilanteen rajoitetun ajan aiheuttamaa stressitilaa on vaikea harjoitella perinteisin opetuskeinoin: Teoriaopetus ei siihen pysty ja simulaatioissa harvoin päästään irti kontrolloidusta ympäristöstä. Pelin avulla voidaan komplikaatioon tuoda monia muuten vaikeasti simuloitavia elementtejä, kuten veren roiskumiset ja potilaan autenttiset reaktiot.

Pelimekaanisesti komplikaatio on lähellä valmistelua. Näkymä on samanlainen ja vaihtoehtoja voi olla enemmän kuin yksi. Peli antaa kuitenkin enemmän palautetta valinnoista: potilas reagoi animaatioilla ja fasilitaattorin kommenteilla (kuva 13).

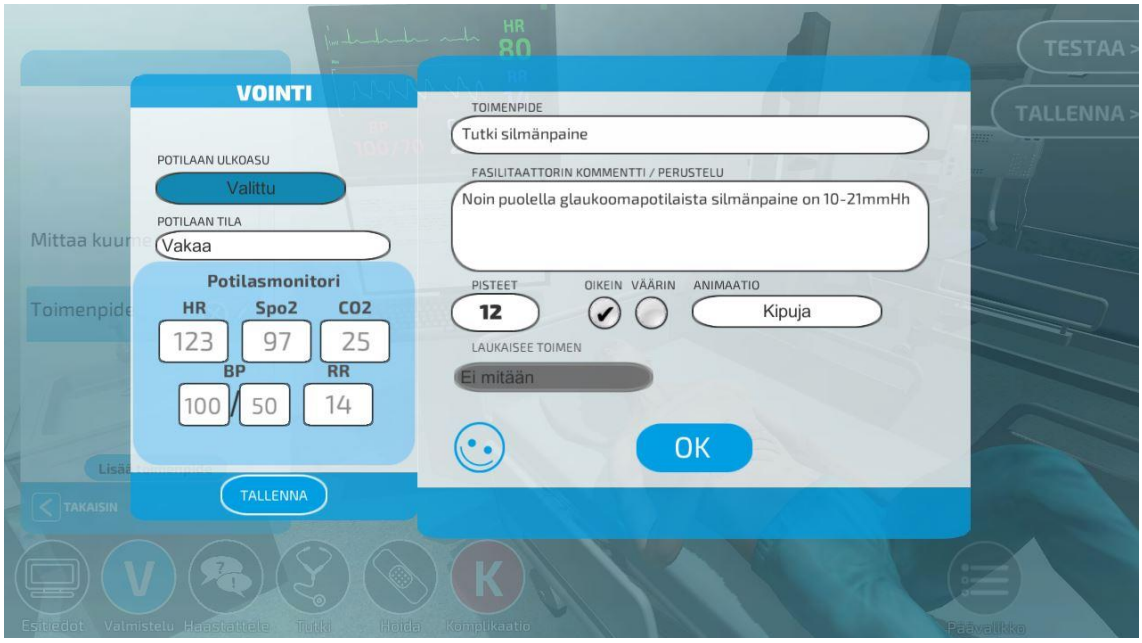

Kuva 13. Potilas menee tajuttomaksi.

## 4.2 Skenaarioeditori

Skenaarioeditori on opettajan työkalu skenaarioiden luomiseen. Editorissa luodaan skenaarioita pelinäkymää vastaavassa näkymässä ja edetään miltei samalla tavalla kuin itse pelissä. Editorista pyrittiin tekemään mahdollisimman intuitiivinen käyttää, ja testausten perusteella moni opettaja osasi käyttää sitä muutaman pelikerran perusteella. Editori toteutettiin osaksi muuta CareMe-peliä Unity Web Playerin sisälle, jotta koko sovellus saatiin yhtenäistettyä niin ulkonäön kuin toiminnan puolesta.

Editorissa opettaja määrittelee tapauksen lähtötilanteen. Hän kirjoittaa oppimistavoitteet, potilastietojärjestelmän tekstin ja säätää potilasmonitorin lähtöarvot. Myös potilaan ulkoasua voi muuttaa. Opettaja pystyy luomaan laajoja hierarkkisia vuoropuheluita ja määrittämään niihin liittyviä animaatioita. Jokaiseen kohtaan pystytään liittämään myös fasilitaattorin ohjausta kommenttien muodossa.

Tutkimusten ja toimenpiteiden luominen on yhtä intuitiivista. Näissäkin kohdissa opettaja pystyy määrittelemään kohtaan liittyvät animaatiot ja fasilitaattorin kommentit. Jokaisessa kohdassa on myös mahdollista säätää potilasmonitorin arvoja, potilaan ulkoasua ja potilaan perusanimaatiota. Jokaisessa kohdassa pystytään määrittämään kohdasta saatavat pisteet ja sitä, onko kohta oikein vai väärin.




Kuva 14. Toimenpiteen luominen skenaarioeditorissa

Skenaarioeditorissa opettajalla on mahdollisuus testata potilasskenaariota lennosta ennen pilveen tallentamista. Napin painalluksella editorityökalut katoavat, ja skenaario alkaa. Skenaarion voi tallentaa myös vajaana palveluun ja jatkaa muokkaamista myöhemmin. Web-pohjaisuus mahdollistaa opettajan työskentelyn niin kotoa kuin kesämökiltä käsin.

# 5 CareMe:n toteutus

## 5.1 Unity Web Player

Käyttäjälle näkyvä osa sovelluksesta on toteutettu Unity3D:llä ja käännetty Unity Web Player -muotoon. Tämä mahdollistaa laajan jakelun ja käytön paikasta riippumatta web-selainten kautta.

Suurin ongelma projektin alkaessa oli pelin käyttöliittymäpainoitteisuus. Unity on ollut pitkään tunnettu heikoista UI-työkaluistaan. Onneksi juuri hieman projektin aloituksen jälkeen julkaistiin Unity 4.6, jossa nämä työkalut olivat päivitetty nykystandardien mukaiseksi.

Käyttöliittymän kehityksessä projektissa käytettiin seuraavaa työnkulkua: teollinen muotoilija suunnittelee ja toteuttaa käyttöliittymäelementin InDesign tai Photoshop-ohjelmistossa ja se tuodaan Unityyn. Unityssä luodaan kuvaelementti tästä elementistä ja se sommitellaan peliin. Tähän kuvaelementtiin lisätään muita elementtejä, mm. erilaisia tekstinsyöttölomakkeita tai painikkeita. Lopuksi minä toteutan käyttöliittymän toiminnallisuuden yhdistämällä elementit sovelluskoodissa.

Toisena isona päänvaivana projektissa olivat yhteydet Web Playeristä taustajärjestelmiin. Toteutustavaksi päätettiin RESTful-arkkitehtuurimalliin perustuva rajapinta, jota voidaan kutsua http-protokollalla Unityn www-luokkaa käyttäen [17].

Muihin suurempiin ongelmiin kuului kuvien lataaminen sovellukseen. Unity Web Player ajetaan eräänlaisessa hiekkalaatikossa tietoturvallisuuden takia. Web Player itsessään ei osaa olla tekemisissä käyttäjän tietokoneen kanssa, eli kuvan lataaminen oli toteutettava eri tavalla.

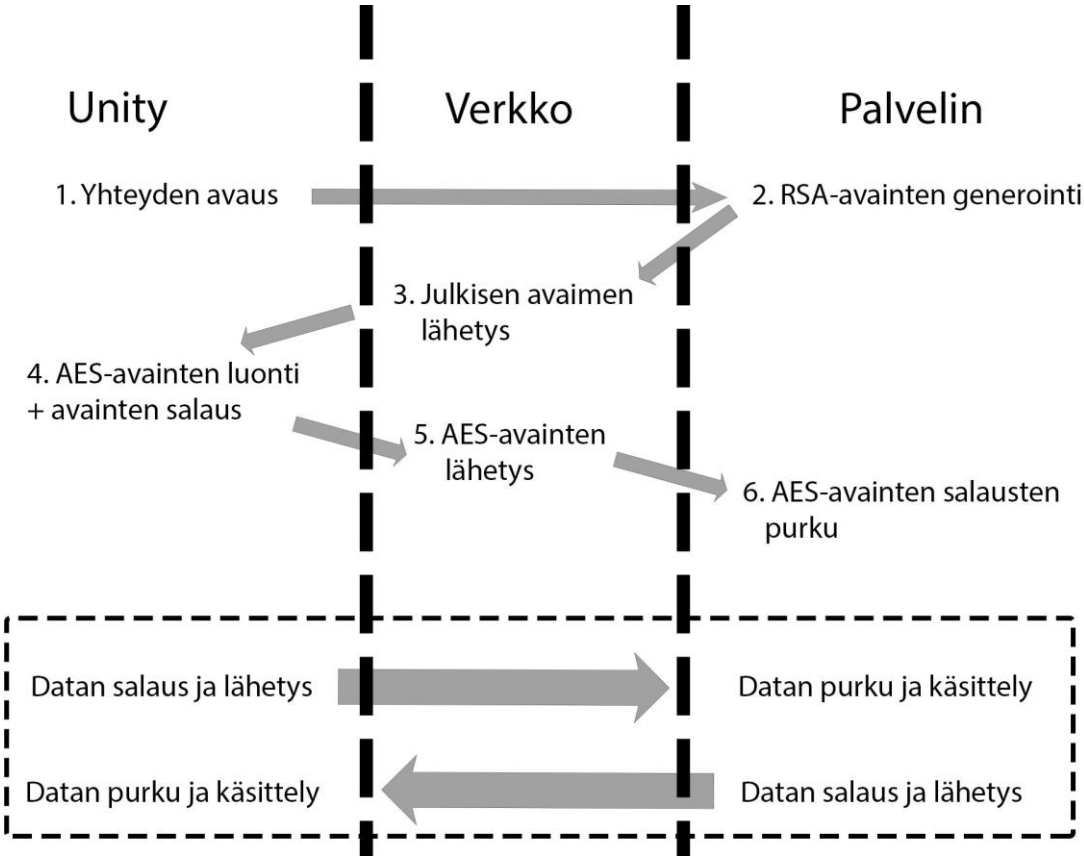
Vaikka Web Player ajetaan rajatussa ympäristössä, osaa se keskustella sen html-sivun kanssa, jossa se sijaitsee. Kuvan latausta varten kutsutaan html-sivulla sijaitsevaa JavaScript-koodia, joka avaa ponnahdusikkunan, jossa sijaitsee latauslomake. Käyttäjä lisää kuvan lomakkeeseen, ja kuva ladataan ensin EC2-palvelimelle, josta se jatkaa matkaa S3-tiedostojärjestelmään. Täältä se ladataan takaisin http:llä Web Playeriin ja näytetään käyttäjälle.

Koska peliä ajetaan pilvipalveluna, tiedostojen koot on pidettävä mahdollisimman pieninä. Isoimpia tiedostoja peliprojektissa ovat 3d-mallit, niihin liittyvät tekstuurit sekä UI-elementtien grafiikka. Insinöörityön kirjoitusvaiheessa pelin käyttäjän tietokoneelle lataaman datan tiedostokoko on pystytty pitämään alle 35 Mb:n kokoisena. Tämä koko tulee kuitenkin pakostakin kasvamaan projektin kasvaessa, kun mallien ja grafiikan määrä lisääntyy.

## 5.2 Autentikointi ja suojaus

Tärkeänä osana jokaista web-sovellusta on käyttäjän tietojen suojaus. Salasanat eivät voi olla tietokannoissa selkokielisenä tekstinä ja liikenne palvelimen ja asiakasohjelman välillä on oltava turvallista. Autentikointi ja suojaus CareMe:ssä on toteutettu RSA- ja AES-algoritmien avulla.

Jokaisen session alussa asiakasohjelma avaa OpenSSL-protokollalla toteutetun yhteyden palvelimeen. Palvelin luo joka sessiolle uudet 2048-bittiset RSA-avaimet ja lähettää asiakkaalle julkisen RSA-avaimen. Asiakasohjelma luo 256-bittiset AES-avaimet ja lähettää ne RSA:lla suojattuina palvelimelle. Palvelin avaa AES-avaimet omalla RSA-avaimellaan. Tästä eteenpäin kaikki liikenne asiakasohjelman ja palvelimen välillä suojataan AES:llä (kuva 15).




Kuva 15. Salauksen toiminta CareMe:ssä

2048-bittisen RSA-salauksen purkaminen nykyteknologialla järkevässä ajassa on mahdotonta [18]. Sama pätee 256-bittiseen AES-salaukseen [19]. Realistisesti mietittynä nykyaikaisesti salaukset ovat niin kehittyneet ja monimutkaiset, että tietoturvaongelmat eivät liity avainten purkuun vaan heikkoon ohjelmistosuunnitteluun tai ohjelmointiin.

Käyttäjien salasanat eivät voi olla tietokannassa selkokielisenä tekstinä. CareMe:ssä salasanat tiivistetään yksisuuntaisella md5-tiivistefunktiolla. Ennen salasanan tiivistystä siihen lisätään vielä erilaisista tiedoista (tietokannasta ja tietokannan ulkopuolelta) luotu "suola" eli satunnainen merkkijono. Tämä poistaa mahdollisuuden käyttää ns. Rainbow Table -tauluja joista voitaisiin eri md5-tiivisteitä tutkimalla selvittää alkuperäinen salasana.

## 5.3 Taustajärjestelmät ja niiden liitäntä peliin

CareMe:n taustajärjestelmät on rakennettu Amazon Web Service -pilvipalveluperheen ympärille. Tämän arkkitehtuurin ytimessä ovat EC2-virtuaalipalvelimet, jotka ohjaavat koko järjestelmää käyttäen apuna muita AWS:n palveluita.

Kun käyttäjä kirjoittaa selaimeensa pelin osoitteen, ottaa selain yhteyttä EC2-palvelimelle. Palvelimella ajettava Apachen web-palvelinsovellus jakelee oikean html-sivun, ja käyttäjä ohjautuu pelisivulle. Tässä vaiheessa haetaan S3-pilvitallennuksesta Unity Web Player -tiedosto. Tiedosto tarvitsee selaimeen pienen lisäosan. Jos lisäosaa ei ole asennettu, tarjotaan käyttäjälle mahdollisuus ladata ja asentaa se.

Kun Web Player käynnistyy, alkaa se lataamaan itse pelin tiedostoja S3-tallennuksesta. Latauksen tilaa kuvataan käyttäjälle latauspalkkina. Ensimmäisessä vaiheessa ladataan ainoastaan kirjautumissivulla tarvittavat komponentit, jotta latausaika pysyisi mahdollisimman lyhyenä.

Pelin ladattua kirjautumissivun komponentit näytetään käyttäjälle kirjautumissivu, jossa kysytään käyttäjätunnusta ja salasanaa. Taustalla sivu alkaa lataamaan seuraavien sivujen komponentteja taustalla. Samalla peli ottaa yhteyttä EC2-palvelimelle ja muodostaa ensin RSA-salatun yhteyden, jonka jälkeen AES-salatun yhteyden.

Käyttäjä syöttää kirjautumistietonsa. ja peli lähettää ne salattuna palvelimelle. EC2-palvelin ottaa tässä vaiheessa yhteyttä RDS-tietokantapalvelimelle ja tekee SQL-kyselyn kirjautumistietojen paikkansapitävyydestä. Jos tiedot eivät täsmää tietokannassa olevien tietojen kanssa, palauttaa palvelin siitä tiedon Web Playerille. Käyttäjälle näytetään ilmoitus. Jos taas tiedot täsmäävät, palautetaan siitä tieto ja ladataan seuraava sivu. Samassa paketissa palautetaan tietokannasta haetut käyttäjätiedot, kuten käyttäjän pistesaldo, ja säilötään ne väliaikaismuistiin salattuna.

Seuraavaksi näytetään valikkosivu. Tältä sivulta löytyvät eri pelattavat potilastapaukset ja painikkeet asetuksiin ja editoriin. Valikkosivun rakenne riippuu käyttäjän statuksesta: onko käyttäjä oppilas vai opettaja?

Valikkosivun latautuessa peli lähettää kyselyn valittavissa olevista potilastapauksista palvelimelle. Palvelin lähettää MongoDB-kyselyn potilastapaustietokantaan ja palauttaa peliin käyttäjätunnukseen liittyvien työtilojen potilastapausten nimet ja id:t listana. Käyttäjä voi olla jäsenenä monessa ryhmässä samaan aikaan (vrt. Moodle).

Käyttäjän valitessa potilastapauksen tekee palvelin kyselyn ja hakee kyseisen tapauksen XML-tiedoston, pakkaa sen ja siihen liittyvät muut tiedot JSON-muotoon ja palauttaa sen peliin. Peli lukee XML-tiedostossa määritellyt esitiedot ja siirtyy pelimoodiin. Itse potilastapauksen aikana peli ei kommunikoi taustajärjestelmien kanssa. Tämä vähentää radikaalisti palvelinten kuormitusta.

XML-tiedostoa käydään läpi hierarkkisessa järjestyksessä. Ensin määritellään kaikki tapaukseen liittyvät asetukset ja lähtötilanne. Tämän jälkeen peli etenee tiedoston mukaan kohdassa 4.1 esitetyn kaavan mukaan.

*Skenaarioeditoriin* siirryttäessä peli pyytää palvelimelta tietoja muokattavissa olevista potilastapauksista. Peli antaa mahdollisuuden luoda uusi tapaus tai muokata olemassa olevaa tapausta. Jos valitaan uusi tapaus, luo peli tyhjän XML-pohjan, joka luetaan editoriin. Samoin toimitaan, kun käyttäjä valitsee muokattavan tapauksen. Tapaus ladataan palvelimelta ja luetaan editoriin.

Editori itsessään on yksinkertaistettuna pelkistetty graafinen XML-editori. Editori luo valmiiksi määritettyjen rakenteiden mukaan potilastapausta yksinkertaisen käyttöliittymän avulla. Testasimme aivan alkutaipaleella editoria lautapelinä, jotta saisimme mahdollisimman selkeän kuvan mitä sillä pitäisi pystyä tekemään (kuva 16).


Kuva 16. Skenaarioeditorin lautapelimalli [20].

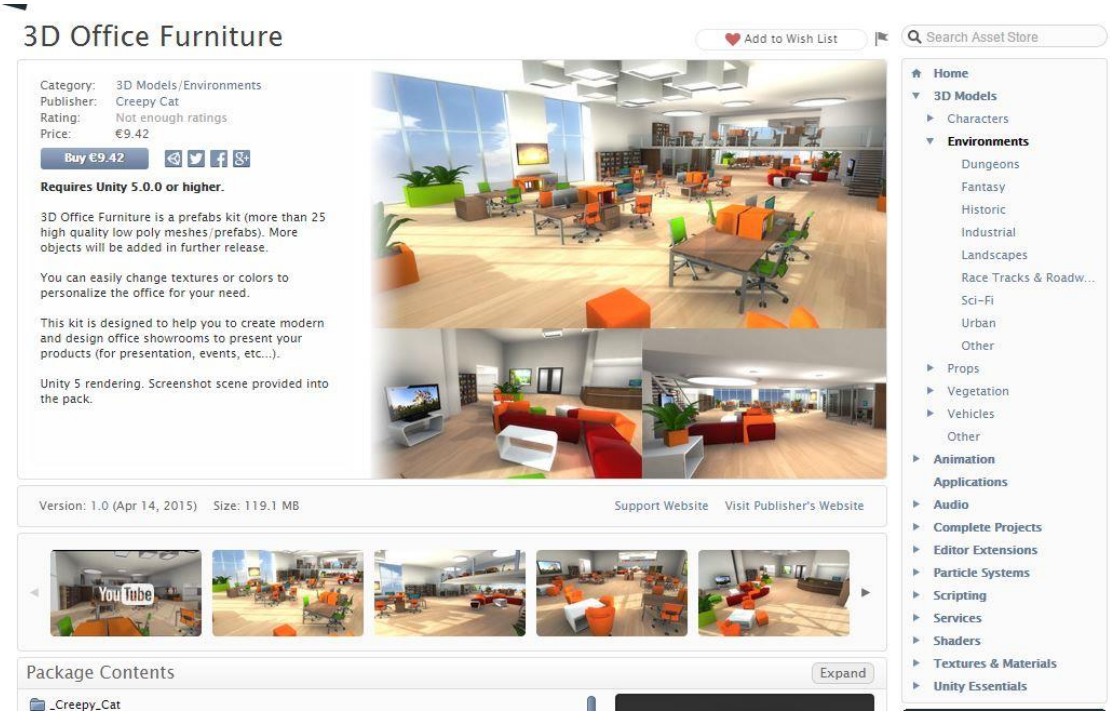
## 5.4 Jatkokehitys

Projekti jatkuu käyttäjähallinnan hiomisella. Miten yksittäisten käyttäjien rekisteröinti toimisi? Miten lisenssijärjestelmä tulisi toimimaan? Näissäkin kysymyksissä on tehokkainta käyttää samaa tekniikkaa kuin skenaarioeditorin suunnittelussa; istua opettajien ja kouluttajien kanssa alas keskustelemaan ja pohtimaan. Jos he eivät näe järjestelmää mukavana ja yksinkertaisena käyttää, eivät he sitä myöskään halua ottaa osaksi työkalupakkiaan.

*Statistiikka ja raportointia* tulee kehittää opettajien ja kouluttajien kanssa yhteistyössä. Mitä tietoja yksittäisistä opiskelijoista halutaan tarkastella? Entä ryhmätasolla? Onko tarpeellista tuottaa graafista dataa? Mahdollisuutta käyttää peliä kokeena pitää myös pohtia.

*Grafiikka* tulee saamaan paljon päivityksiä kehityksen jatkuessa. Peliin tullaan luomaan useita eri potilasmalleja ja niille erilaisia tekstuureja, joita yhdistelemällä pystytään potilaitten ulkoasu pitämään mahdollisimman monimuotoisena. Sama koskee myös animaatioita. Mallien kehityksessä tulee pyrkiä mahdollisimman paljon uudelleenkäytettävyyteen. Mitä enemmän mallien animaatioita voidaan käyttää ristiin, sitä vähemmän työtä ja tallennustilaa ne tulevat viemään.

*Ympäristöjä* tullaan kehittämään lisää. Insinöörityön kirjoitushetkellä pelissä on vain demovaiheessa oleva vuodeosasto. Vuodeosaston jatkokehityksen jälkeen seuraavana suunnitelmissa olisivat koti- ja leikkaussaliympäristöt. Myös ambulanssiympäristö on mietinnässä. Osassa näistä ympäristöistä on mahdollista käyttää Unity Asset Store –palvelua. Jos kuukauden työmäärän tarvitsevan mallinnetun ympäristön voi ostaa alle 10 €:n hintaan (kuva 17), onko järkevää tehdä kaikkea itse?




Kuva 17. Toimistoympäristö Unity Asset Storessa

Lisääntynyt tekstuuri-, malli- ja ympäristömäärä tarkoittavat myös kasvavia tiedostokokoja. Ei ole kohtuullista olettaa käyttäjän pystyvän tai jaksavan ladata pitkälti yli 100 Mb:n kokoisia tiedostoja joka kerta, kun hän kirjautuu peliin. Tämä ei myöskään ole palvelinresurssien tehokasta käyttöä. Tätä varten jatkossa on suunniteltava ja toteutettava järjestelmä lisätiedostojen latausta varten.

Pelin alussa ladattava paketti tulisi olla pieni ja sisältää vain oleelliset tekstuurit ja mallit valikkoihin ja perustapauksiin. Näissäkin tekstuureissa tulisi käyttää mahdollisimman suurta pakkausta ja pientä kuvakokoa. Kun käyttäjä lähtee pelaamaan potilasskenaariota, peli osaisi hakea palvelimelta juuri kyseisessä skenaariossa käytettävät mallit ja tekstuurit ja ladata niistä ensin pienikokoisimmat ja heikkolaatuisimmat versiot. Samalla kun pelaaja lukee potilastietojärjestelmän esitietotekstejä, hakisi peli taustalla parempilaatuisia tekstuureja ja latauksen loputtua vaihtaisi niihin. Näin saavutetaan mahdollisimman sulava pelikokemus maksimaalisella graafisella ilmeellä.

*Pilvipalveluiden lisäkehitystarve* tulee selkeytymään ohjelmiston laajemman käyttöönoton jälkeen. Palvelinmäärää voidaan joutua kasvattamaan ja kuormantasausta säätämään. Amazon Web Service –palveluiden käytössä tämä kaikki toimii yksinkertaisimmillaan parilla klikkauksella niin virtuaalipalvelimien kuin tietokantapalvelimien tapauksissa.

*Pelimekaanisten ominaisuuksien* jatkokehitys- ja lisäystarve tulee suureksi osaksi asiakkailta. Asiakas voi tarvita juuri heidän käyttöönsä elintärkeän ominaisuuden, jota emme ole edes miettineet. Jatkokehityksessä tulemme ainakin lisäämään mahdollisuuden harjoitella erilaisten laitteiden käyttöä.

Laitteiden käyttö ja koulutus tulisi sitoa sulavaksi osaksi potilasskenaariota ja oppimispolkua. Tämä mahdollistaisi paremmin eri koulutushaarojen spesifin opiskelun. Röntgenhoitaja voisi harjoitella röntgenlaitteiston käyttöä ja anestesiahoitaja anestesialaitteiden toimintaa. Tämä kaikki tapahtuisi oikeassa kontekstissa potilasskenaarion sisällä luoden yhtenäisen oppimiskokonaisuuden.

Eri työvaiheiden priorisointia on toivottu opettajien käyttäjätestauksissa.

*Mobiiliversio* pelistä on myös kehitteillä. Unityn parhaita ominaisuuksia on mahdollisuus kääntää sovellus eri alustoille muutamalla napin painalluksella. Pelin käyttöliittymän suunnittelussa ja kehityksessä on pidetty koko ajan mielessä kosketusnäytön mahdollisuus. Käyttöliittymäelementit ovat isoja ja selkeitä. Pienillä muutoksilla pelistä tulisi erittäin hyvin toimiva pienemmälläkin näytöllä. Insinöörityön kirjoitusvaiheessa peliä oli testattu Android-tabletilla.

Koska kommunikaatio taustajärjestelmien kanssa toimii http:n yli, ei taustajärjestelmiä ole tarvetta muokata erikseen mobiiliyhteensopiviksi. Isoimpia töitä joutuisi tekemään optimointipuolella. Vaikka mobiililaitteet nykypäivänä ovat tehokkaita, graafisesta ilmeestä voi joutua karsimaan. Reaaliaikaiset heittovarjot ja läpinäkyvät elementit ovat tableteilla vielä isoja ongelmia.

# 6 Yhteenveto

Insinöörityössä esiteltiin CareMe:ssä käytettäviä työkaluja ja tekniikoita niin asiakasohjelmisto- kuin pilvipalvelupuolella. Työssä käytiin läpi pelin toimintaa käyttäjätasolla ja tutustuttiin pelin taustajärjestelmien logiikkaan. Käytännön osuudessa kehitettiin CareMe-pelin pelillisyyttä ja pelimekaanikoita sekä kehitettiin ulkoasua lähemmäs nykystandardeja.

Graafinen ulkoasu saivat uuden potilaan, animaatioiden ja käyttöliittymä-elementtien kautta ison päivityksen. Todella iso hyöty ulkoasun kehityksessä oli Unity 4.6-version mukana tulleilla UI-työkaluilla.

Taustajärjestelmien kehityksessä tutustuin moniin entuudestaan tuntemattomiin palveluihin ja työkaluihin. Erityisen paljon opin Amazonin pilvipalveluista. Tämän projektin perusteella tulen käyttämään niitä tulevissakin projekteissa.

Olen erittäin tyytyväinen siihen, mitä tiimimme sai aikaan puolessa vuodessa. Aloitimme projektin miltei tyhjältä pöydältä ja onnistuimme rakentamaan toimivan pelin, jota on jopa mukava pelata. Käyttäjätutkimuksissa tulee aina positiivisena yllätyksenä, kuinka hyvän vastaanoton CareMe saa.

Tiimityöskentelytaitoni ovat kehittyneet projektin aikana paljon. Opin erityisen paljon eri ihmisten, myös itseni, optimaalisista työskentelytavoista. Ohjelmistotuotannossa ei voi odottaa saavan joka päivä toimistoaikoihin yhtä paljon aikaan. Pitkäksi venyneet illat deadlinen lähestyessä kuuluvat projektityöskentelyyn ohjelmistojen kehityksessä.

## Lähteet


1. State of Gaming 2013. Verkkodokumentti. <www.spilgames.com/press/2013-state-online-gaming-report-released-spil-games>. Luettu 13.4.2015.

2. Evidence for striatal dopamine release during a video game. Verkkodokumentti. <www.nrc-iol.org/cores/mialab/fijc/files/2002/120402_koepp_nature_1998.pdf>. Luettu 8.4.2015.

3. Hallikainen Juhana, Väisänen Olli, 2007. Simulaatio-opetus ensihoidossa. Verkkodokumentti. <www.finnanest.fi/files/hallikainen_simulaatio.pdf>. Luettu 3.4.2015.

4. Joutsen Mikko, Jakobson Joonas, 2014. Sairaanhoitajaopiskelijoiden kokemuksia oppimispelin pelattavuudesta. Verkkodokumentti. <www.theseus.fi/handle/10024/78972>. Luettu 2.4.2015.

5. Unity3D Multiplatform. Verkkodokumentti. <unity3d.com/unity/multiplatform>. Luettu 23.2.2015.

6. Unity's Animation System. Verkkodokumentti. <docs.unity3d.com/Manual/AnimationOverview.html>. Luettu 11.4.2015.

7. Play Framework Documentation. Verkkodokumentti. <www.playframework.com/documentation/2.3.x/Home>. Luettu 3.3.2015.

8. Amazon Web Services. Verkkodokumentti. <aws.amazon.com/>. Luettu 8.4.2015.

9. The MongoDB 2.6 Manual. Verkkodokumentti. <docs.mongodb.org/manual/>. Luettu 23.2.2015.

10. RAID – Redundant array of independent disks. Verkkodokumentti. <www.webopedia.com/TERM/R/RAID.html>. Luettu 23.2.2015.

11. Ohjelmointiputka. MySQL-opas. Verkkodokumentti. <www.ohjelmointiputka.net/oppaat/opas.php?tunnus=mysqlphp01>. Luettu 13.4.2015.

12. Apache HTTPD Wiki. Verkkodokumentti. <wiki.apache.org/httpd/FAQ>. Luettu 27.3.2015.

13. GIT vs SVN. Verkkodokumentti. <www.codeforest.net/git-vs-svn>. Luettu 10.3.2015.

14. Trunk vs. HEAD in Version Control System. Verkkodokumentti. <garygregory.wordpress.com/2011/02/03/trunk-vs-head-in-version-control-systems/>. Luettu 5.3.2015.

15. A Systematic approach to the acutely ill patient. Verkkodokumentti. <www.resus.org.uk/pages/alsABCDE.htm>. Luettu 7.4.2015.

16. Critical Care Nurse Journal. Verkkodokumentti. <ccn.aacnjournals.org/cgi/content-nw/full/24/5/19/T4>. Luettu 1.4.2015.

17. Learn Rest. Verkkodokumentti. <rest.elkstein.org/>. Luettu 23.2.2015.

18. The Math Behind Estimations to Break a 2048-bit Certificate. Verkkodokumentti. <www.digicert.com/TimeTravel/math.htm>. Luettu 12.4.2015.

19. How secure is AES against brute force attacks? Verkkodokumentti. <www.eetimes.com/document.asp?doc_id=1279619>. Luettu 12.4.2015.

20. Nylund Saku, Electria Metropolia. 2014.

**Muistiinpanoja sairaanhoito-opiskelijoiden tapaamisesta**

**5.9.2014**

*Onko liian tylsää jos pelissä käytettävät tasot ovat nimeltään: perehtyvä, suoriutuva, pätevä ja taitava taso? Osaamisen tason kuvaamisessa voisi käyttää huumoria, keventäisi. Liian totista ja tylsää jos kaikki termit ovat ammatillisia.*

*Mikä on palkitsevaa? Pisteytys, tähditys. Pääsy uusille tasoille. Kilpailu toista vastaan; ”Kuka on viikon paras sairaanhoitaja?”*

*Visuaalinen palaute voisi tulla vasta tehtävän suorittamisen jälkeen. Ei niin oleellista että potilaassa tapahtuu visuaalisia merkkejä voinnin muutoksesta tehtävän suorituksen aikana.*
*Sanallinenkin palaute riittää kun tekee virheen. Palautteessa kerrotaan väärän toimenpiteen seuraus potilaalle (perustelu miksi vastaus on väärä).*
*Vinkkien/neuvojen ei tarvitse olla kovin tarkkoja ja yksityiskohtaisia.*
*Potilaan ei tulisi hukkua ruutujen taakse.*
*Potilaan tiedot saisi avattua milloin vain tehtävän suorittamisen aikana.*
*Animoitaviksi asioiksi riittää potilas ja hoitovälineet.*
*Tehtävää suorittavan sairaanhoitajan käsillä tekemisen ei tarvitse näkyä. Pelaajahahmo on ylipäätään ylimääräinen.*
*Riittää esim. että intubaatio putki ilmestyy paikalleen.*

*Onko pelaajien välinen kommunikointi tarpeellista?*
*Pelaajat voisivat kommunikoida ryhmän jäsenten kesken mikrofonin välityksellä toisilleen pelin aikana.*

*Opiskelijat olivat käyttäneet pelissä lokia, jossa näkyy potilastapauksen aikana suoritetut toimenpiteet. Lokissa näkyy myös väärät vastaukset, mutta ei mitään selitystä miksi se on väärä. Lokin symboli oikeassa yläkulmassa.*

*Toive: ”Ei pelkkää tekstiä”*

*Kuvat helpottavat hahmottamaan tilannetta. Esim. Yhden toimenpiteen keissiin voisi toimia kuvan raahaus. Helpommalla tasolla kuvat voisi toimia. Esimerkkitapaus: haavanhoito.*
*Kuvat pitävät kiinnostusta yllä.*

*Eritasoisia virheitä; ”keltainen kortti”, ”punainen kortti”*

*Monitorin tulisi olla helposti näkyvässä muodossa. Saisi helposti havainnoitavaan kokoon halutessa.*

*Ongelmia:*
*Puhekuplat jäävät liian pitkäksi aikaa leijumaan, ei pysty vaikuttamaan niiden sulkeutumiseen.*
*Peli ei anna vinkkejä.*
*Peli jumittuu yhdenkin väärän valinnan seurauksesta. Tehtävä on aloitettava alusta. Virheet ovat saman arvoisia.*